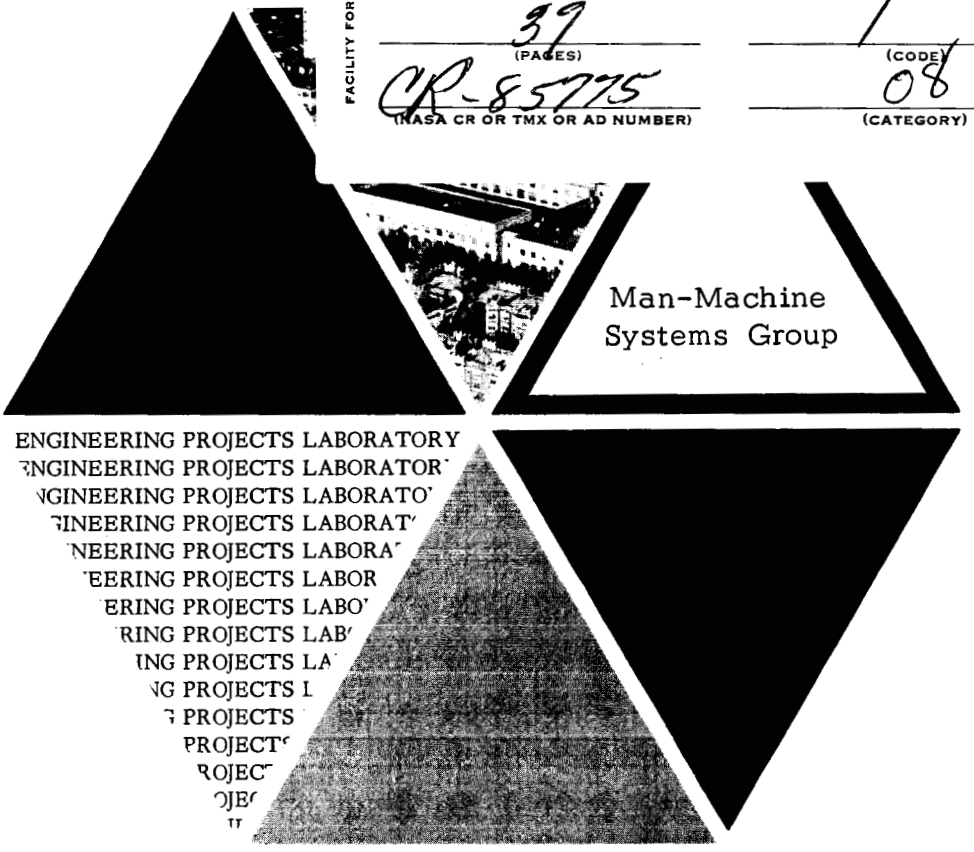
PROGRESS REPORT
October 1, 1966 - March 31, 1967
NASA Grant NsG 107-61

MEASUREMENT AND DISPLAY OF CONTROL INFORMATION
(Remote Manipulation and Manual Control)

Thomas B. Sheridan
William R. Ferrell

DSR 70283-1

Engineering Projects Laboratory
Department of Mechanical Engineering
Massachusetts Institute of Technology



DSR 70283-1

PROGRESS REPORT

Oct. 1, 1966 - March 31, 1967

NASA Grant NsG 107-61

MEASUREMENT AND DISPLAY OF CONTROL INFORMATION

(Remote Manipulation and Manual Control)

by

Thomas B. Sheridan
William R. Ferrell

TABLE OF CONTENTS

I. REMOTE MANIPULATION

A. A Compiler Language for Human Supervisory Control of a Manipulator

-- David J. Barber

Current effort is directed toward evaluating a computer language developed specifically for supervisory control of a manipulator. This language, called MANTRAN, is essentially a compiler which accepts English commands from the typewriter keyboard and generates the detailed motor commands called for. This is quite similar to a FORTRAN compiler which accepts English directions for algebraic manipulations.

There are two new features which should extend the flexibility of previously developed command structures: 1) the ability to reference commands relative to the hands current position, and 2) the ability to give conditional branching statements which the computer executes depending on remotely generated end conditions.

Regarding (1) above, there is no direct feedback of the hand's position to the computer because of the use of stepping motors. The computer merely maintains a list of count registers specifying how many increments each motor has been moved. From this list it is easy for the computer to perform the trigonometric calculations necessary to move in a particular direction or orientation relative to its current position. These calculations as of now take no account of which of several alternative trajectories may be best, but rather arbitrarily choose one. "Best" could be defined as the shortest path, or the one imparting the least momentum to the arm, etc.

Conditional statements [(2) above] are commands as:

1. MOVE LEFT 1000
   UNTIL A TOUCH ANYTHING
   IF MOVE CONDITION FULFILLED, DO 2
   IF A FULFILLED, HELP

2. MOVE FORWARD 10 and RIGHT 1000.
UNTIL B TOUCH ANYTHING
IF MOVE CONDITION FULFILLED, DO 1
IF B FULFILLED, HELP

Underlined portions are typed by the computer while the rest is input by the operator.

The above type of command is typed while the manipulator arm waits quiescent. This particular example is a set of directions to search a plane defined by the hand's current position until the touch sensors on the hand are activated. This program would be executed by the operator's typing

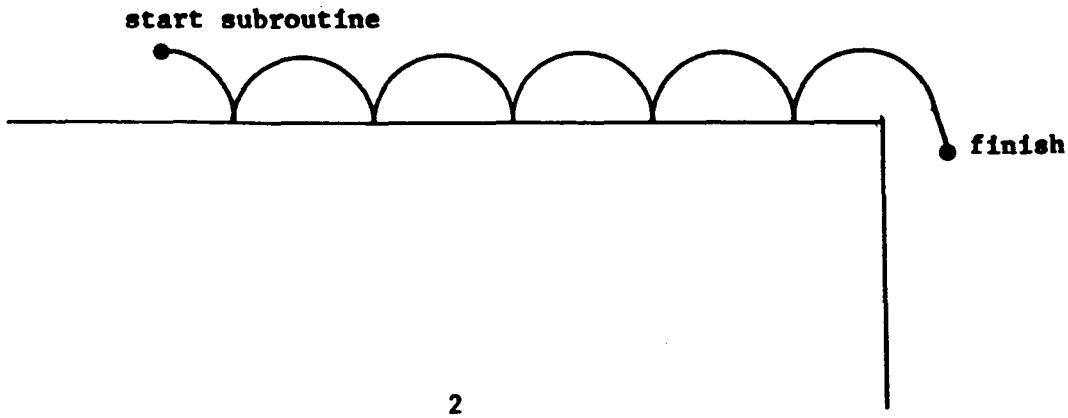
DO 1.

The hand would then sweep left and right while moving slowly forward until it touched something.

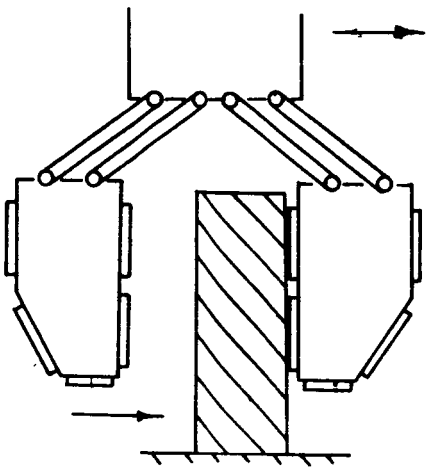
At that time, motion would stop, the computer would print out its current position and the current state of the touch sensors, and wait for HELP from the operator in the form of new instructions.

Because one statement can call for the execution of another, commands can be strung together and called as subroutines merely by DO ing the 1st statement.

Several subroutines have been written which seem useful in manipulation. One is a series of pokes along a surface until a discontinuity is detected, as illustrated below.



A second is a routine which closes the jaws and stops each one separately as it touches. The jaw motor closes both jaws symmetrically with respect to the manipulator wrist: thus a subroutine must be provided after one jaw touches, which translates the wrist and closes the jaws simultaneously keeping the touching jaw stationary. This is shown below.



wrist translates left while jaws close effecting a net translation of left jaw only.

A final refinement would be subroutines with variable arguments which are typed whenever the subroutine is called.

B. State Space Models of Remote Manipulation Tasks

-- Daniel E. Whitney

Remote manipulation is difficult enough if the operator is close

to his work, because there is meager feedback and the apparatus is clumsy and hard to control. Add to this a significant time delay and efficient manipulation becomes almost impossible. The current research attempts to equip the manipulator with some intelligence of its own (a small computer) so that it can evaluate feedback and receive fairly general commands from a far distant human operator. This would relieve the operator of the grubby details whose delayed transmission or receipt causes so many difficulties. To this end, an Optimal Control approach to programming the small computer leads to a state space model of manipulation tasks. Dynamic Programming finds the shortest paths in the state space and these paths are interpreted as work descriptions which are directly intelligible to the manipulator's prime movers.

1. Our goal is to enable the human operator to give commands which are goal oriented (even natural language statements) and equip the local manipulator brain with ability to translate these into a sequence of sub-goal oriented commands directly intelligible to the prime movers, having the property that the sub-goals lead to the stated goal.

2. If we had a state vector and state space for remote manipulation tasks, then the operator could specify a configuration of manipulated objects as a state in the state space; desired changes in this configuration would correspond to changes in the state vector. Unit transitions in this space should correspond to elementary commands $\underline{u}$ which one or two prime movers could accomplish in one simple action. A sequence of commands might read: move 1" left, open, move 1" left, close, move 1" right,... Optimal control theory can find paths between specified states in this space to minimize some criterion function J. Such paths will then consist of a sequence of commands which, when read like a work description, tell the manipulator how to do the task.

What could such a state vector look like? It must show situations we are interested in, like possible jaw positions combined with object positions, etc. If a situation we want appears in state space, we can describe it to the computer and the computer can find a path to get there.

Consider a grid of points each representing a whole situation (state). Adjacent points are connected by lines if there exists a command $\underline{u}$ in the set of allowed commands which will change one situation into the neighboring one. A detail of the environment may disallow some changes, which means some lines are missing. We can make the commands uniform all over and charge for them. Then we get state vector $\underline{S}$ transitions according to:

$$\underline{S}_{k+1} = \underline{S}_k + \underline{u}_k, \quad \underline{u}_k \text{ chosen from } \left\{ \begin{array}{l} \underline{u}_a \\ \underline{u}_b \\ \underline{u}_c \\ \cdot \\ \cdot \\ \underline{u}_n \end{array} \right\} = \text{allowed set of commands}$$

$$\underline{S}(0) = \underline{S}_0$$

$$\underline{S}(n) = \underline{S}_n \qquad (n \text{ to be determined})$$

$$J = \sum_{i=1}^{n} C_{i-1,\,i} \qquad \text{where } c > 0 \text{ and is a cost of a command}$$

Thus we have a state space model of remote manipulation tasks complete with cost function and equation of motion, where our optimization procedure finds $\underline{u}_1, \underline{u}_2 \ldots \underline{u}_n$ minimizing J.

3. The minimal state vector probably must contain the coordinates of the jaws. (Other possibilities include position and orientation of the object.) Consider Figure 1 where a one dimensional physical space is shown. The state space is represented in Figure 2. Thus,

$$S = \left\{ \begin{matrix} x_J \\ J \end{matrix} \right\} \text{ with } x_J = 1,2,\ldots,5 \text{ and } J = 0,1 \text{ (open, closed)}$$

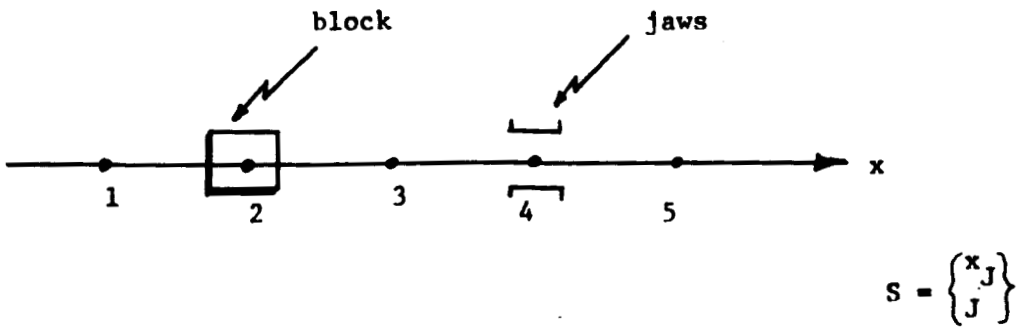


Fig. 1 Physical space

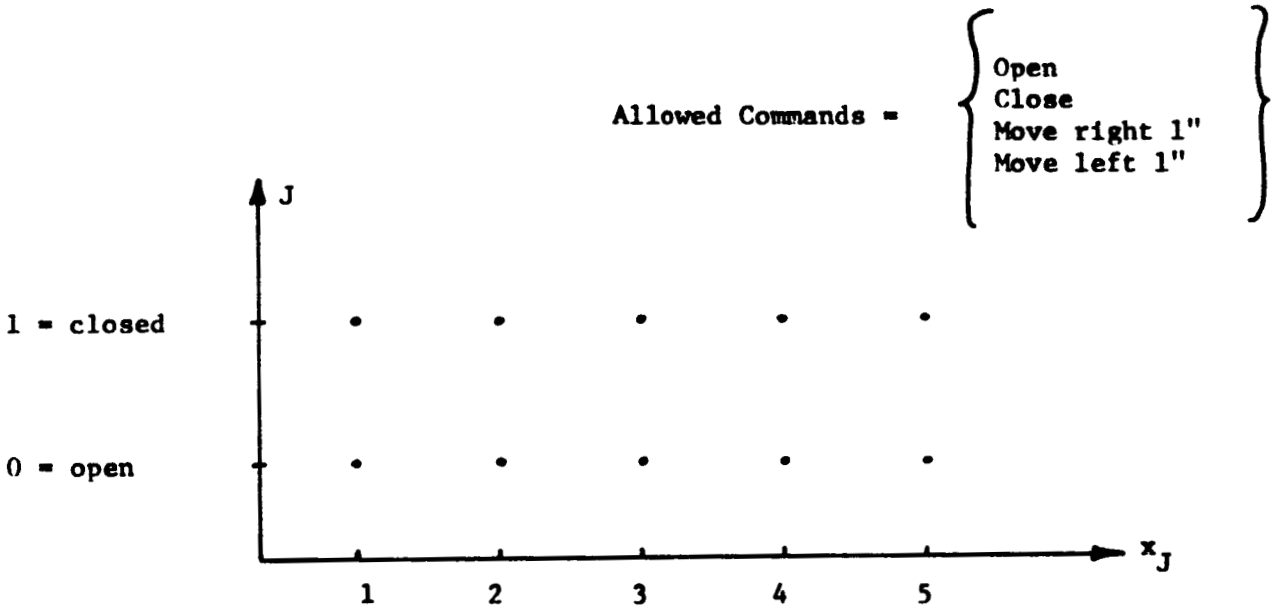


Fig. 2 A state space corresponding to Fig. 1

Say an object is in 2, jaws in 4, closed. Then the system occupies state $\begin{Bmatrix}4\\1\end{Bmatrix}$. We command: move jaws to 1. The computer can easily translate this and determine that the desired terminal state is $S = \begin{Bmatrix}1\\1\end{Bmatrix}$. So we demonstrate the ability to interpret some kinds of desires and express them in the state space framework.

Now, how does the computer figure out how to get the jaws to 1? We denote the allowed transitions in state space by lines between the states (Fig. 3).

we can close jaws at any x.

ability to slide along. Blocked at 2 in this by object.

Next we assign some length or cost to each allowed transition, basing these costs on what we would like done in general, but still not dictating the details of the solution. We deem it inappropriate for the jaws to move from position to position while open unless necessary. Thus we have the situation represented by Figure 4. Then we tell the computer to find the shortest path from $\begin{Bmatrix}4\\1\end{Bmatrix}$ to $\begin{Bmatrix}1\\1\end{Bmatrix}$. It is marked $\leftarrow$. Notice that by naming the states on the solution path in order, we obtain a work description of the solution:

move one unit left
open
move one unit left
"
close
done

Simple computer routines can carry out these commands one at a time. Therefore the translation from the operator's command to the computer's work description is complete.

We could also ask that the object in 2 be grasped: $S_i = \begin{Bmatrix}4\\1\end{Bmatrix}$, $S_f = \begin{Bmatrix}2\\1\end{Bmatrix}$

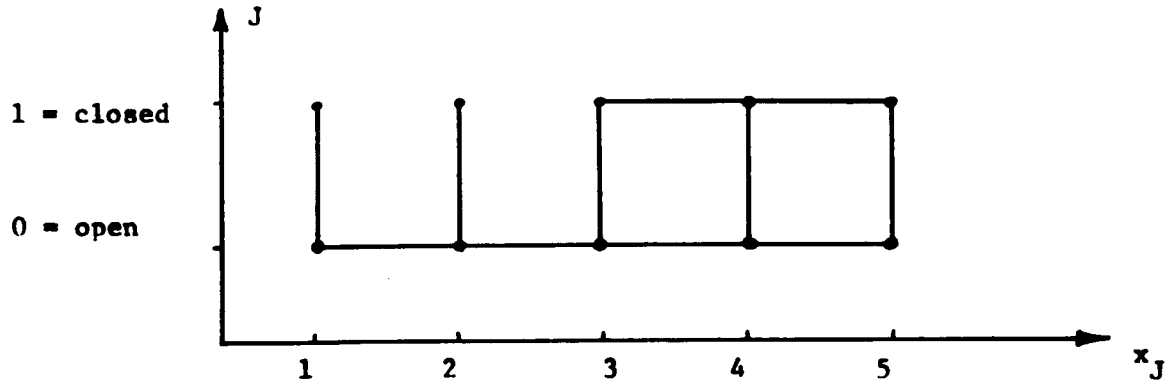


Fig. 3 Allowed Transitions of S based on Figure 1

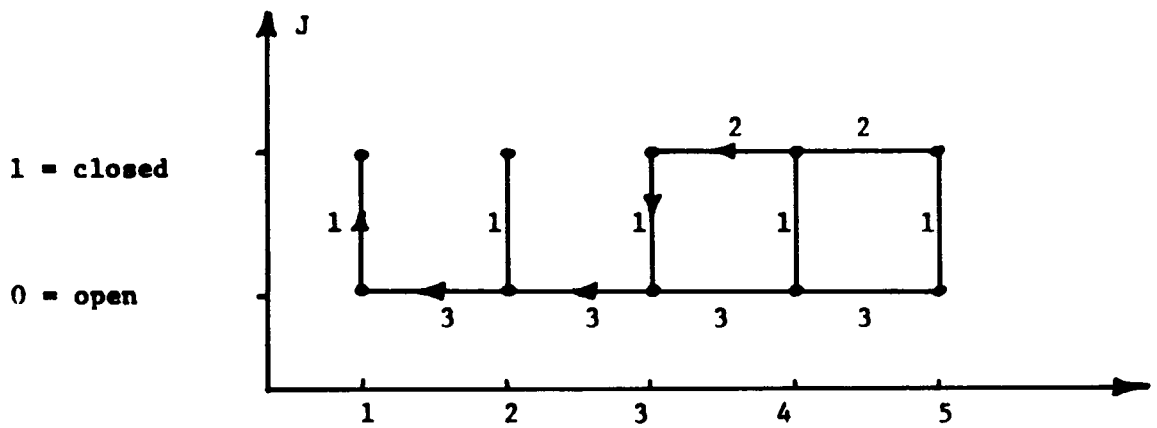


Fig. 4 A shortest path from $S = \begin{Bmatrix} 4 \\ 1 \end{Bmatrix}$ to $S = \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$

This will be done automatically and sensibly:

move one left
open
move one left
close
done

A problem arises if we consider the distinction between closed empty and closed full. One can't straddle another object if closed full. So we can expand the state vector so that J = 0, 1, -1 (Fig. 5). Thus new situations → new states → new commands.

The missing transitions in Figure 5 are consistent with the presence of an object in 2. Now we can say: Take the object in 2 to 5: $S_i = \begin{Bmatrix} 4 \\ 1 \end{Bmatrix}$ $S_f = \begin{Bmatrix} 5 \\ -1 \end{Bmatrix}$ butthis only makes sense if there is an object in 4 and the state space reflects this. The computer would then have to modify the state space at the completion of the task to show an object at h. Then the command "go to 3, closed", would be translated $S_i = \begin{Bmatrix} 5 \\ -1 \end{Bmatrix}$, $S_f = \begin{Bmatrix} 3 \\ 1 \end{Bmatrix}$ and in this way we could let go of the object and back away. But a lot of this modification of the state space could be avoided if the position of the object were a state variable. Again, new states, new commands.

To do this, let $S = \begin{Bmatrix} \underline{x} \\ \underline{y} \\ J \end{Bmatrix}$, $\underline{x}$ = jaw position, $\underline{y}$ = object position, J as before. Now we have a situation as indicated in Figure 6. Now we can say: take object from 3 to 5 and leave jaws closed in 3:

$S_i = \begin{Bmatrix} 4 \\ 3 \\ 1 \end{Bmatrix}$, $S_f = \begin{Bmatrix} 3 \\ 5 \\ 1 \end{Bmatrix}$. This state space need not be modified as the object is moved around. The left graph indicates that jaws and object cannot occupy the same position while jaws are closed empty. Both the left and center graphs indicate that the object cannot change position by itself but the jaws can. The right graph shows that the object is moved by bringing the jaws to the object's position, closing, and moving them together. The solution automatically obtained is:

Allowed Commands = { Open, Close, Grasp, Release, Move left 1", Move right 1" }

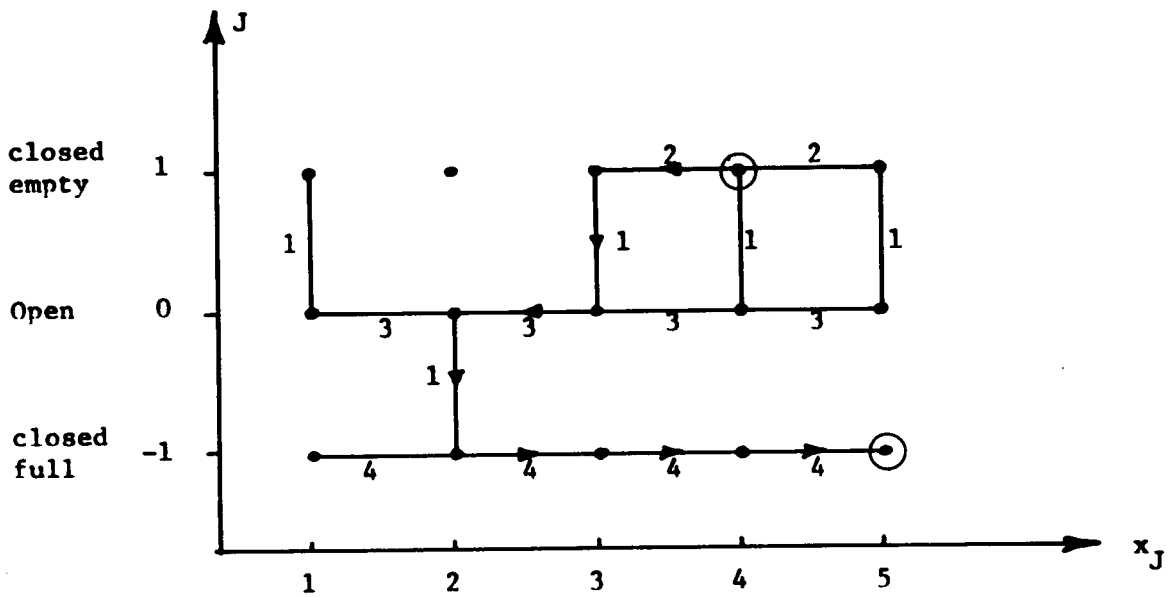


Figure 5

A shortest path from $S = \begin{Bmatrix} 4 \\ 1 \end{Bmatrix}$ to $S = \begin{Bmatrix} 5 \\ -1 \end{Bmatrix}$

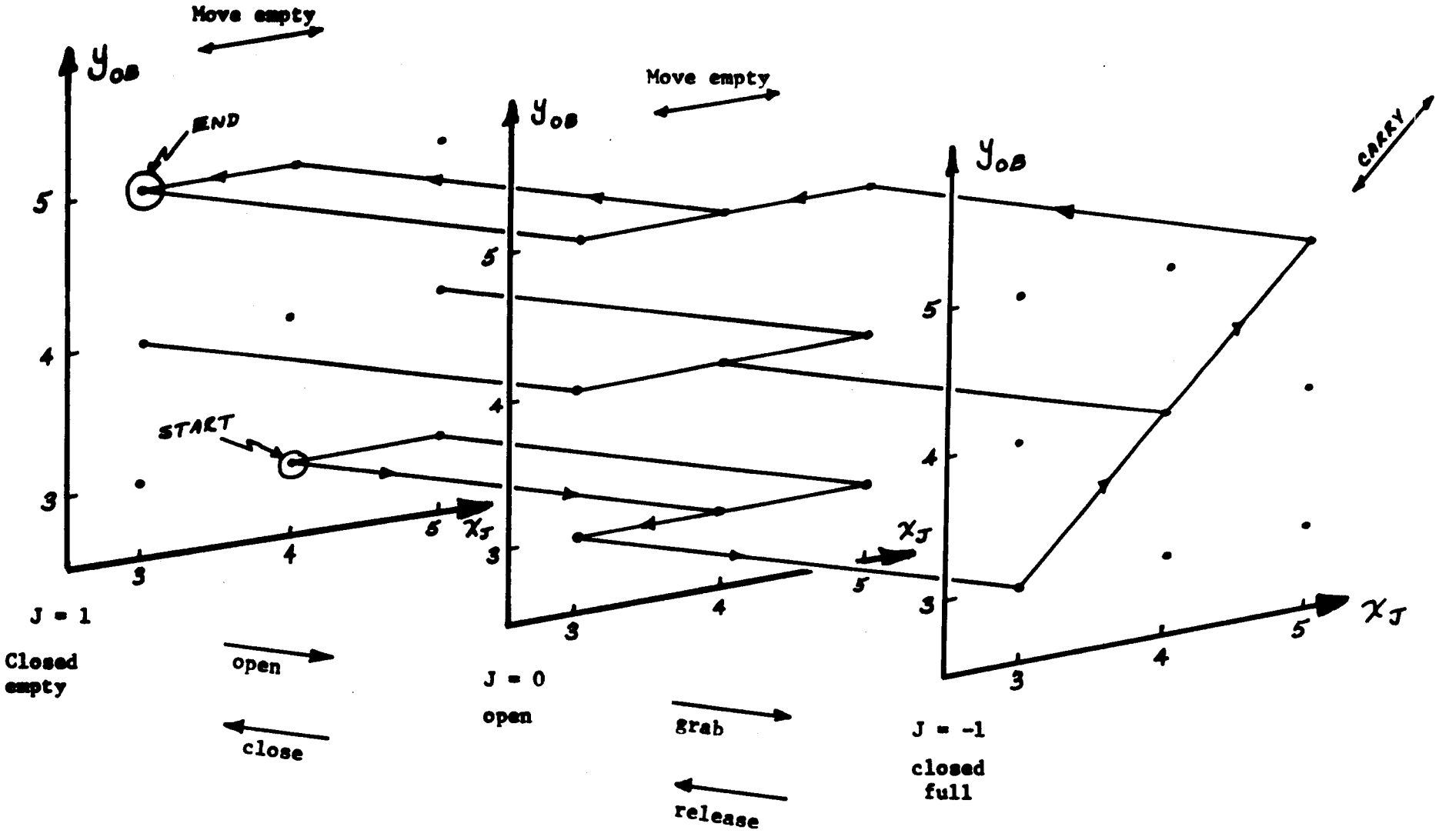


Fig. 6 Trajectory in state space for complete task.

```
   open
move left 1
   grasp
carry right 2
   release
move left 1
   close
move left 1
   done
```

Note that Figure 5 is a portion of Figure 6 for the restriction that y = 2. Figure 5 therefore is a two-dimensional cross-section through the three-dimensional Figure 6. The progression from Figure 3 to 6 increases the power of the input language. We can say more and it can do more, ultimately all of it automatically.

There are serious dimensional problems with this method. We have a one-dimensional physical space with 5 points and to handle only one object and jaws we have 3 two-dimensional graphs with 75 locations. Enormous state spaces will result from larger dimensioned physical spaces with finer quantization. Say we have one object, jaws and three-dimensional physical space with 10 points per axis: each segment is a six-dimensional graph of 10^6 points and there are 3 of them. Such difficulties arise in all similar problems.

What else is there to manipulation besides moving things around?

a) Apply forces and torques (plug in a plug)

b) Fit things together (nut on bolt)

c) The trouble is that people have some trouble doing these things. Can we expect a machine to do them?

d) They require enormously detailed and complex feedback of pressure, shear, and contact and kinesthetic data. We do not know how to describe the accomplishment of these tasks, so how can we tell a machine? Certain investigators suggest having enough artificial intelligence so that the machine will learn how to do these things, perhaps by our criticizing its efforts. This seems a long way off.

C. Evaluation of Optical Touch Sensor - Jack Krafchick

In order to use effectively the remote manipulator for various manual tasks, some form of tactile information should be available to the operator to supplement position information (i.e. visual or graphical). To this end, a prototype optical touch sensing device (OTS) was designed by Strickler (1) to be used in conjunction with an AMF model 8 manipulator.

Figure 7 is a schematic representation of the method used to present this tactile information to the operator. It involves the visual interpretation of optical patterns generated at the remote jaws. The pattern generator forms one of the bearing (sensing) surfaces of the jaws. When it is pressed against an object in a grasping maneuver, regions of contact may be ascertained, and force levels estimated. These optical changes are viewed by the operator through a visual system consisting of a television camera and monitor, and a coherent fiber optic bundle. The fiber bundle transmits the optical pattern from the pattern generator to the camera, and provides the flexibility necessary to follow the motions of the remote jaws.

The final design of the OTS evolved after several other design techniques were abandoned. The latter included photoelastic stress pattern generators and Moire pattern generators. The final design involved a reflected grid technique whereby the reflection of an array or grid is viewed in a flexible mirror. As seen from Figure 7a, the plexiglas prism acts as a mounting stage for the other components of the pattern generator. The transparent rubber is actually a silicone potting compound known as Sylegard 184 (by Dow-Corning). This compound is water clear and is capable of 100% elongation. The flexible mirror is made of a piece of half-silvered mylar, 0.007 in. thick. The grid is a checkerboard array of black squares (5 per inch) printed on cellophane. A light source mounted behind a translucent screen illuminates the grid, and produces an image on the flexible mirror. The T.V. camera views the grid pattern on the flexible mirror via the fiber-optic bundle. The grid is coarse enough so that the individual components

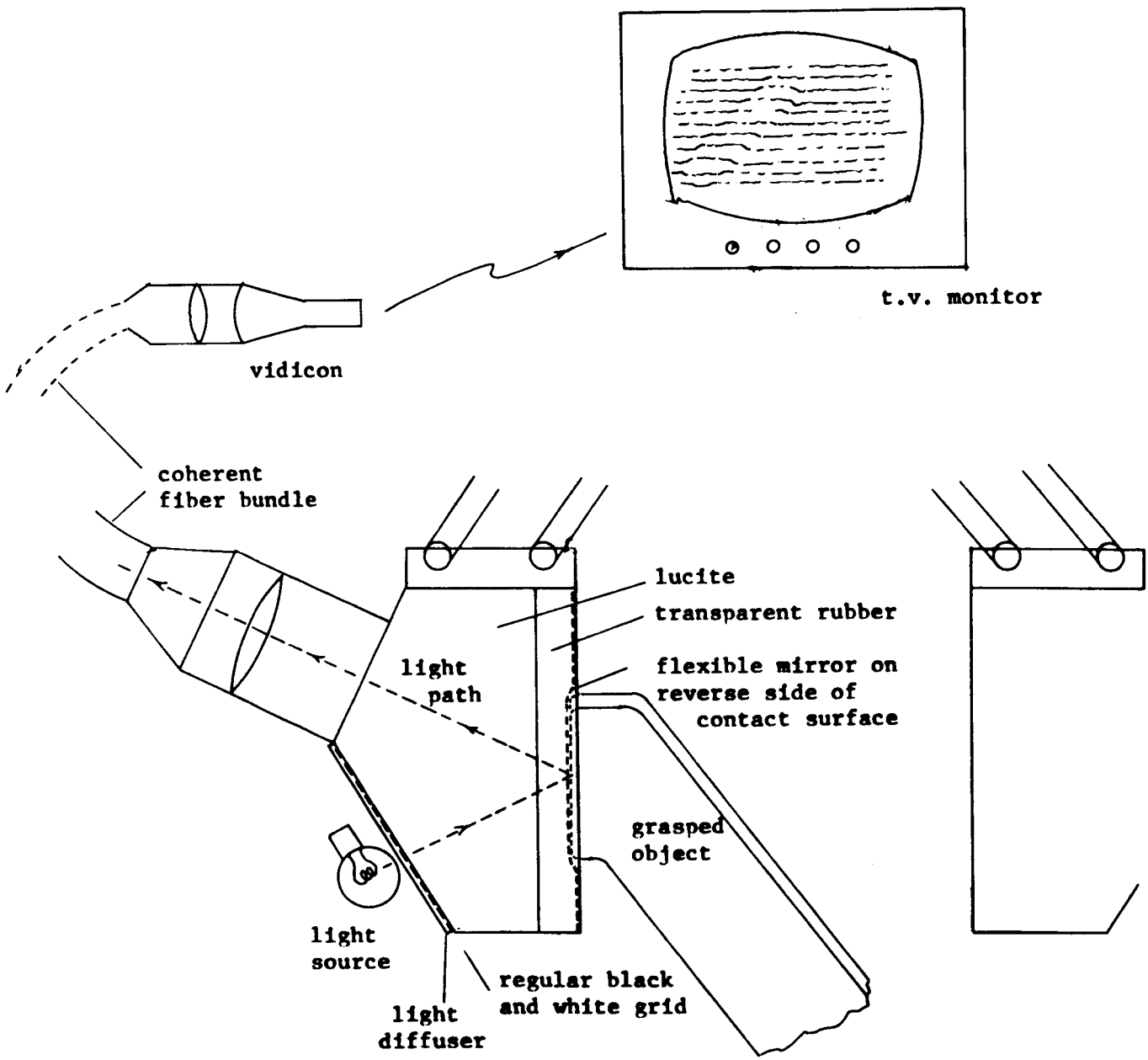


Fig. 7a Schematic of optical touch sensor

(squares) are easily distinguished. Distortions of the grid's reflection (due to mirror distortions) are easily seen as discontinuties in the symmetry of the array.

In order to evaluate the characteristics and usefulness of the OTS, it is being used in conjunction with a remote manipulation task wherein all other forms of kinesthetic or tactile feedback to the operator were eliminated. To achieve this end, a second AMF model 8 manipulator is being used as a master controller. This master manipulator is outfitted with position transducers for each seven degrees of freedom, and is interfaced with the PDP-8 digital computer.

The slave manipulator is driven by the PDP-8 computer and is position controlled by stepping motors for each seven degrees of freedom. The OTS is mounted on the slave manipulator, while the T.V. display is presented to an operator who controls the master manipulator.

The computer routine for synchronous coupling of the master and slave manipulators is accomplished by assigning a position coordinate to the transducer signal for each degree of freedom. The program logic keeps track of current and previous positions. By determining if the difference between current and previous position is greater than a specified tolerance, a motor pulse is issued to increment a particular stepping motor in the proper direction. This synchronous coupling of the master and slave manipulators is accomplished in an open loop, such that if synchrony is ever lost (i.e. stepping motor slips a pulse) the master and slave remain out of phase. Initialization and repositioning are required to regain synchrony.

The stepping motors have typical torque-speed characteristics such that output torque decreases as pulse frequency increases. Therefore, under usual conditions a velocity lag must be endured in order to achieve sufficient torque output from the stepping motors. This velocity lag is under direct control of the computer program, and may be changed at will,--but only as a trade-off with available torque.

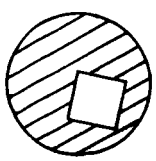
The usefulness of the OTS to the human operator is associated with such tasks as shape discrimination, determination of object orientation with respect to remote jaws, and determination of gripping force of jaws. Thus several tasks could be designed which would demonstrate the effectiveness (or not) of the OTS with respect to such tasks.

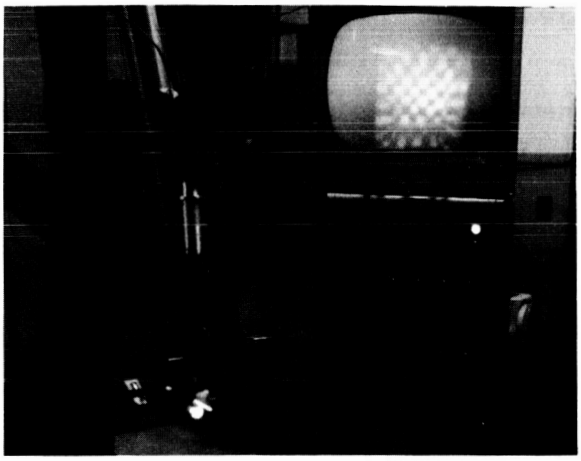
Shape Discrimination

The ability to discriminate between geometrical shapes is being determined by considering a given set of shapes (triangle, square, octagon, etc.); the operator is asked to identify each shape. The only information which is supplied to the operator is via the OTS. The first set of identifications is to be made without any prior information concerning the given shapes in the set. After the first trial, the operator (subject) is told what shapes constitute the set, and then the identification procedure is repeated. Fig. 7b illustrates the operator's display of several shapes.

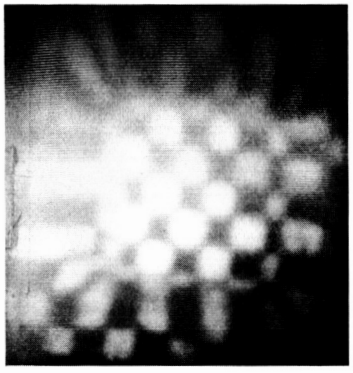
For each shape, the percentage of correct identificationsis recorded, and the entire task repeated on several subjects. A parameter closely associated with shape identification is the resolution obtainable with the OTS. For a specified grid size and a specified elasticity of the transparent rubber bearing surface, the resolution is determined by

1) two point discrimination
2) ability to discriminate existence of a smaller depression within a larger circular cross-sectional shape.

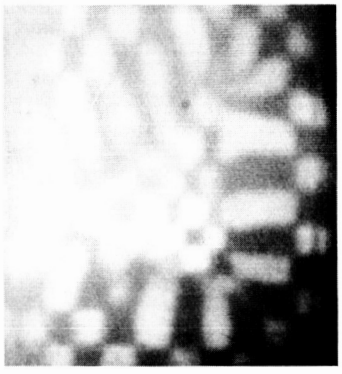




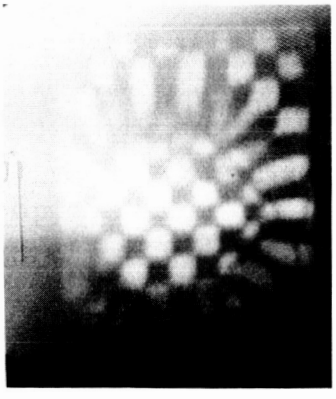
overview showing manipulator,
(light source is bright spot at bottom)



square



triangle



hexagon

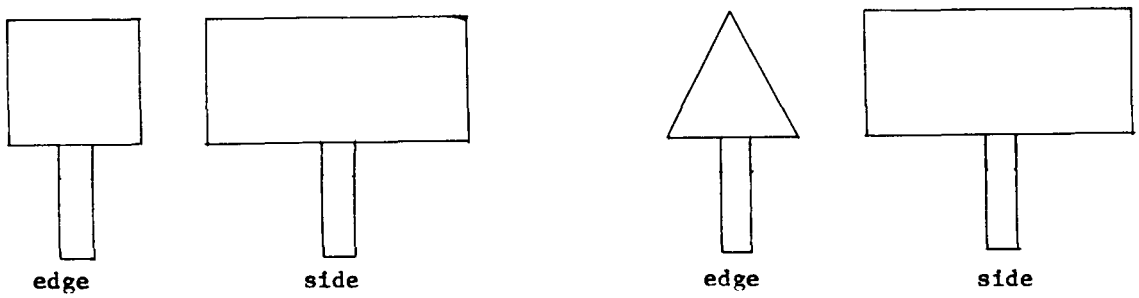
circular cross section
with concentric hole

Fig. 7b Examples of geometrical patterns as transmitted by the optical touch sensor through the coherent fiber bundle, and TV camera and viewed on the TV monitor.

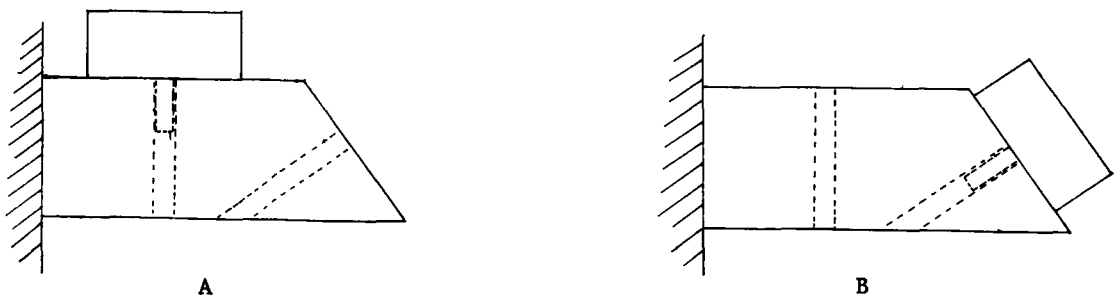
Orientation Discrimination

In any manipulation task, the orientation of an object with respect to the jaws becomes important. To determine if the OTS can convey sufficient information concerning object orientation to the operator the following task is employed.

An object with an asymmetric cross-section (isosceles triangle, rectangle, and trapezoid) has a short protrusion from one face. The task is to move the object from position A



to position B, knowing only the location of A and B from visual information. The orientation of the object in the jaws is obscured from the view of the operator since the object is smaller than the jaws. Therefore, the only information oncerning object orientation in the jaws comes from the OTS.



If the relative orientation of the object and jaw remain fixed, then the operator need only reposition the manipulator and insert the object into B. But if the relative orientation is changed while the object is in the jaw, then the operator can compensate for this (from information supplied by the OTS) before insertion into B can be effected. It is assumed that this change in orientation will be controlled by the experimenter. The performance parameter is time to effect insertion at B after reorientation. As a comparison, it may prove useful to measure completion time 1) without reorientation, 2) without OTS and with reorientation and 3) without OTS and without reorientation.

Estimation of Gripping Force of Jaw

The ability to calibrate and quantify the point by point normal gripping force of the jaw is also being studied.

Preliminary Experimental Results

A. Shape Determination:

Several subjects were asked to identify the shapes listed in Table 1 by using the OTS mounted on the remote manipulator. The tabulated results are the number of times each shape was presented, and the number of correct identifications. The order of presentation was random, and the shapes that were more difficult to identify were presented more often.

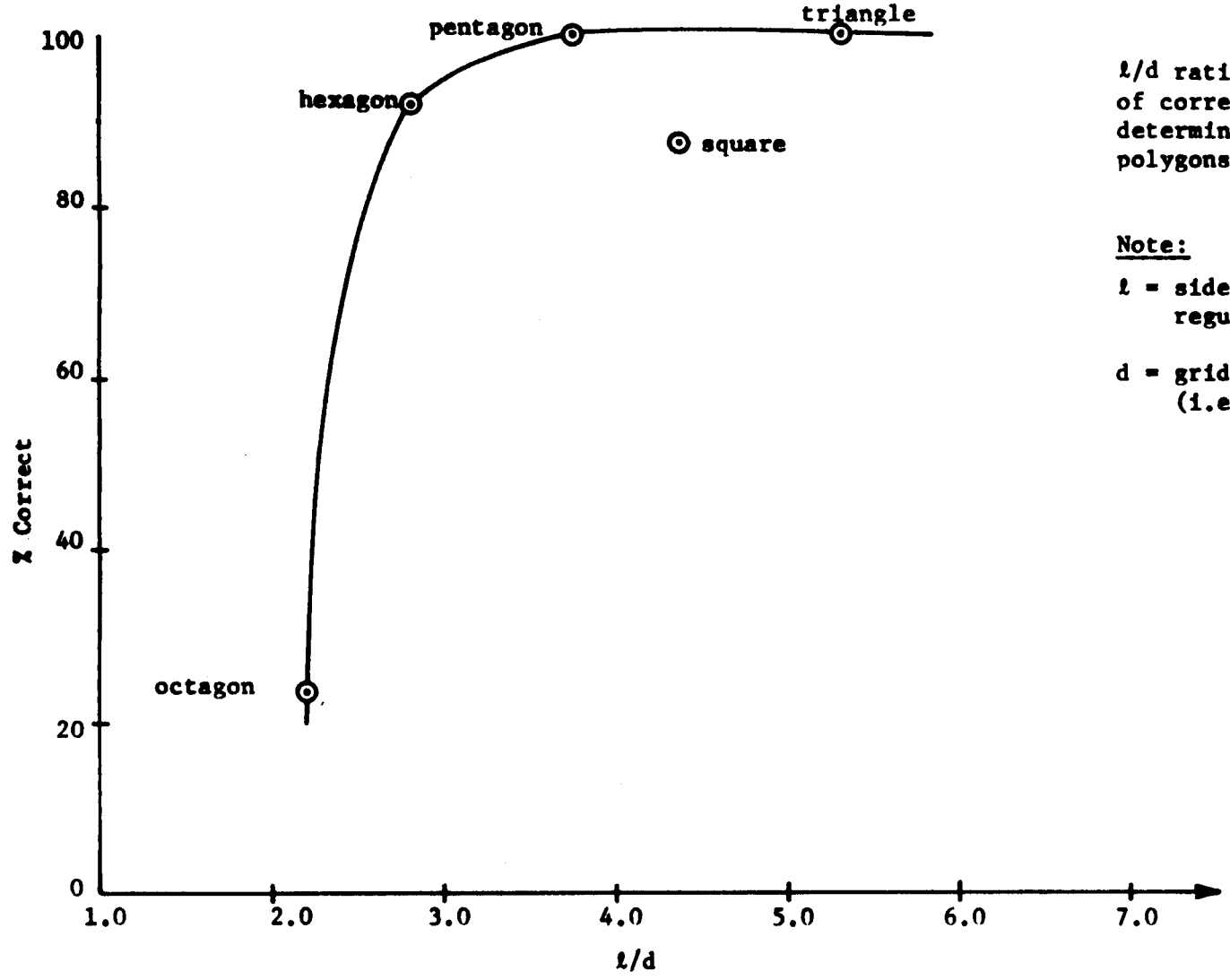
Table II shows the summary of the totals for the three subjects. Considering only those shapes which are classified as regular polygons, a dimensionless parameter can be formed relating the length of a polygon side to the grid size. If ℓ = length of a polygon side, and d = grid size, the ratio ℓ/d becomes a function of the resolution obtainable with the OTS. Figure 7c shows a plot of the ℓ/d ratio for the shapes used versus the percentage of correct determinations as shown in Table II. Hopefully the dimensionless ℓ/d ratio would apply for any regular polygon and any grid size. The data point referenced as a square is unreliable since the number of trials was small.

Subject →	W.		C.		V.	
Shape	No. of Pres.	No. Correct	No. of Pres.	No. Correct	No. of Pres.	No. Correct
Triangle	2	2	3	3	2	2
Square	2	2	3	3	3	2
Pentagon	4	3	3	3	3	3
Hexagon	4	3	5	5	4	4
Octagon	4	0	6	2	4	1
Circle	5	4	6	2	4	1
Parallelogram	2	2	3	3	2	2
Rectangle	1	1	3	3	1	1

Table I - Individual Trials

Shape	Total No. of Pres.	Total No. Correct	% Correct
Triangle	7	7	100
Square	8	7	87.5
Pentagon	9	9	100
Hexagon	13	12	92.3
Octagon	14	3	21.4
Circle	15	7	46.7
Parallelogram	7	7	100
Rectangle	5	5	100

Table II - Total of Three Subjects



ℓ/d ratio vs. percentage of correct shape determinations for regular polygons.

Note:

ℓ = side length of a regular polygon

d = grid size (i.e. 10/inch → d = 0.1")

Fig. 7c - Effective resolution of the OTS.

B. Object Orientation

From the preliminary data on shape determination, the orientation of a rectangle or isosceles triangle is easily determined with the OTS. The manipulation task described in the previous section was attempted by each subject, but because of backlash in the grar trains and finite stepping motor increments on the remote manipulator, no one was able successfully to complete the task.

Discussion and Conclusions

The ability to identify elementary shapes is easily accomplished by using the OTS only. Based on the preliminary data of Figure 7c, a minimum ℓ/d ratio of about 3.5 will permit accurate identification of any regular polygon. In addition to the static information regarding shape and orientation as found in Fig. 7b, the performance of an actual task also conveys a certain amount of secondary information arising from the formation of the distortion pattern as the compressive force on the object is changed. The dynamic picture allows better identification.

II. CONTINUOUS MANUAL CONTROL

A. Time-Optimal Control of a Second Order System by a Human Operator -- Duncan C. Miller

Experiments are designed to investigate the ability of the human operator to bring a second order system to rest in minimum time. In these experiments, the human operator is provided with a three-state hand control (+U, 0, -U), one of the four controlled systems listed below, and one of the four displays listed below. The controlled system is released from one of a set of initial conditions, and the human operator is required to operate the hand controller in such a manner as to bring the system to zero position and zero velocity (within a small tolerance) in minimum time.

The four systems used have transfer functions:

$$(1)\quad \frac{x}{u} = \frac{K}{s^2}$$

$$(2)\quad \frac{x}{u} = \frac{K}{s^2 + .25}$$

$$(3)\quad \frac{x}{u} = \frac{K}{s^2 + s + .25}$$

$$(4)\quad \frac{x}{u} = \frac{K}{s^2 + .25s + .25}$$

For each of these, the time-optimum switch curve can be calculated fairly easily, and the human operator's performance can be measured against optimal.

The four displays used are as described in our last progress report (Sept. 30, 1966). They are: (1) the position (x) only, a one dimensional display; (2) the phase plane, position vs. velocity; (3) the phase plane with the addition of two predicted trajectories, one indicating the future path of the system with the present input, and the other indicating the future path with the negative of the present input; (4) the switch curve, the phase plane display upon which is superimposed the optimum switch curve.

Three subjects are being run on each of the four systems with each of the four displays. In each case, the subjects' performances are measured by two criteria: (1) the total time taken to complete a series of runs as compared to the total optimum time, and (2) the distribution of switch points used by the subject as compared to the optimum switch curve.

The data so far indicate that subjects do very well with the switch curve and predictor displays, worse with the phase plane display and worst with the position only display. The switch points used cluster quite closely around the optimum curve in the first two displays just mentioned and are more scattered with the second two.

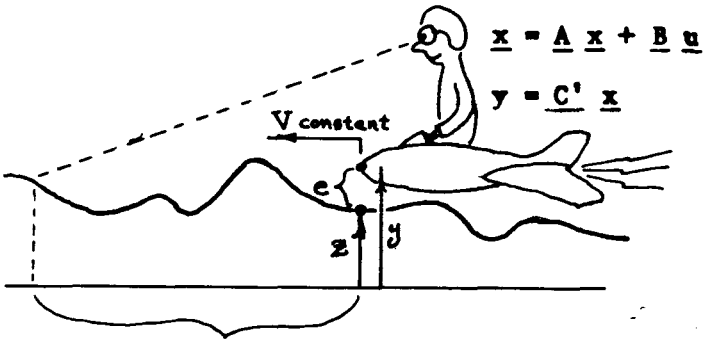
It seems encouraging that the subjects do almost as well with the predictor display as when the optimum switch curves are displayed. This may have an important practical implication. Switch curves are fairly difficult to calculate; usually, digital computation is required. If the system parameters used in calculating the switch curve are incorrect, or if one of the parameters changes during a control task, performance can be seriously degraded. On the other hand, predicted trajectories can be generated by analog means, and require only a fast-time model of the system. This model can be updated fairly easily to compensate for any changes in system parameters and performance will not be degraded.

During the next few months the investigation will be extended to include other cost criteria and perhaps some constraints on the system. In each of these cases the analytical optimum switch curves are much more difficult to calculate. For this reason it can be shown that the human operator can do nearly as well with a predictor display of some kind as with the switch curve display; then these results might have valuable practical implications in terms of reduced hardware and software complexity.

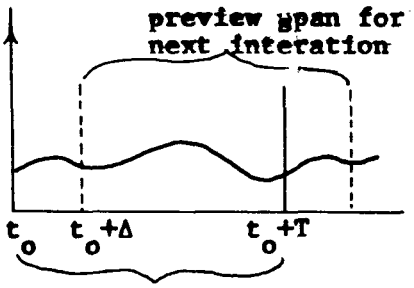
B. Formal Analysis of Preview Control -- Harry Vickers

Continuing with his study of one-dimensional tracking with preview as reported in a previous report, H. Vickers has derived expressions for the optimum linear preview controller.

Consider the problem of controlling a general linear plant $\underline{\dot{x}}(t) = \underline{A}\,\underline{x}(t) + \underline{B}\,\underline{u}(t)$ (time invariant n^{th} order system, $\underline{x}$ = state vector), with the control $\underline{u}(t)$, having a preview of the input $\underline{z}(t)$ for T sec. ahead. The output of the system is $\underline{y}(t) = \underline{C}^T\,\underline{x}(t)$. Conceptually we can think of a vehicle moving down the road at a fixed speed and its driver (or control system) can see ahead.



field of view
0 to T seconds



preview span for this iteration

The hypothesis is that the way to minimize overall cost, given only this much preview, is to take that control action which minimizes cost over the immediate preview span. Then after proceeding along this trajectory for a time Δ, ($\Delta \ll T$), we again look at the new preview span and compute a new trajectory based on our current state and this new input information.

So for $t_0 < t < (t_0 + \Delta)$ (during one cycle) we have the following problem:

$$\text{minimize } \underline{J} = \int_{t_0}^{t_0+T} \underline{L}(\underline{e}, \underline{u}, t)\, dt$$

subject to constraints

$$\dot{\underline{x}} = \underline{A}\,\underline{x} + \underline{B}\,\underline{u}\ ,\ \underline{x}(0) = \underline{x}_0$$

$$\underline{y}(t) = \underline{C}^T\,\underline{x}(t)$$

$$\underline{e}(t) = \underline{z}(t) - \underline{y}(t) = \text{error (may be vector or scalar)}$$

We have taken L to be the quadratic form:

$$\underline{L} = \underline{e}^T \underline{E}\, \underline{e} + \underline{u}^T \underline{M}\, \underline{u} \quad , \underline{E} \text{ \& } \underline{M} \text{ positive definite}$$

the solution to the problem is:

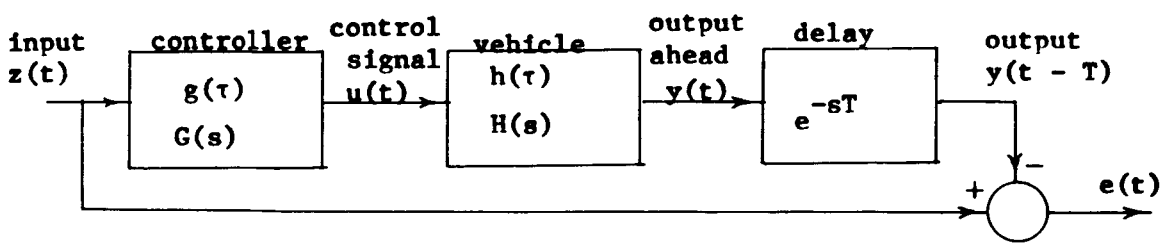
$$\underline{u}(t_0 + \zeta) = \underline{M}^{-1}\, \underline{B}^T\, \underline{\phi}^T\, (-\zeta, 0) \int_{t_0 + \zeta}^{t_0 + T} \underline{\phi}^T(\sigma, t_o)\, \underline{C}\, \underline{E}\, \underline{e}(\sigma)\, d\sigma$$

$$\text{for } 0 < \zeta < \Delta$$

where $\underline{\phi}\ (t, t_0) = e^{\underline{A}}\ (t - t_0)$ = the plant transmission matrix. The form of this solution is very close to the form derived for the two-time-scale model described in the next section.

This problem can also be approached from the classical Weiner-Hopf viewpoint. In this case we assume that the (scalar) control u(t) can be expressed as a linear operation on the input over all available time ($-\infty$ to t + T) i.e. Therefore

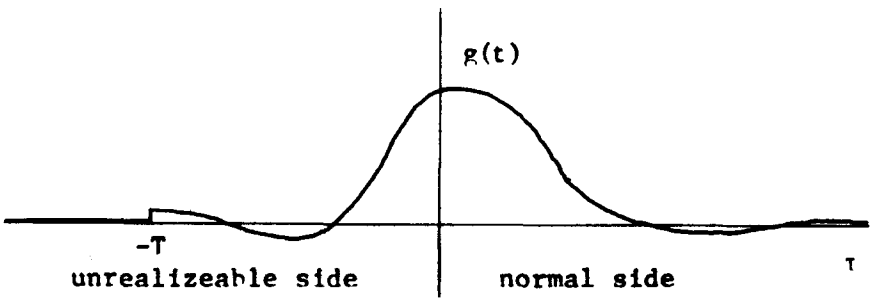
$$u(t) = \int_{-T}^{\infty} g(\tau)\, z(t - \tau)\, d\tau$$



We minimize $K_e^2\ tu^2$ where the ( ) indicates ensemble average. Thus the inputs statistics must be stationary and known. The solution for the optimum

$$G(j\omega) = \frac{e^{j\omega t}}{[KH(-j\omega)\ H(j\omega) + 1)\ \Phi_{zz}(\omega)]_L} \int_0^{\infty} dt\ e^{-j\omega t}\ \frac{1}{2\pi\ j} \int_{-j\infty}^{j\infty} \frac{[KH(-p)\ \Phi_{zz}(p) e^{-pT} e^{pt}]dp}{[KH(-p)\ H(p) + 1)\ \Phi_{zz}(p)]_R}$$

where Φ_{zz} (ω) is the power spectrum of the input, $H(j\omega)$ is the plant transfer function, $G(j\omega)$ is the previewer transfer function, and where $[\ ]_L$ & $[\ ]_R$ indicate left and right half plane spectral factorization. The weighting function that would correspond to this for a typical input spectrum and a second order plant would look something like this:



An attempt is being made to look at the connection between the two approaches which in detail appear different but in purpose are almost the same. Also there appears to be some connection between these models and the recent experiments by W.R. Ferrell on information transformation rate as a function of preview, but a computer program is needed to do the calculations. We expect to have this by the end of the summer.

Vickers is also trying to extend the concept of preview control from the one-dimension tracking to a more comprehensive multi-dimensional terminal control task with a general cost function defined on the space. In this case the controller has to pick its own path, not simply follow a given path. The problem has direct application to remote manipulation, low-flying radar avoidance aircraft or studies of the blind where one knows the desired end state (possibly through inertial navigation) but not all of the obstacles (areas of high cost) along the way. One can only "see" the obstacles near

the present state. The more preview available, the easier is the task, at least conceptually; in the extreme case of complete preview we have a standard optimal control problem. In the limited preview case the solution must be somewhat heuristic, possibly using expected overall cost (if this can be computed) to make the local decisions.

In addition to problems that must be solved in this way (due to a real lack of information) there is reason to believe that many complex tasks with full vision (such as a complicated manipulation task in many dimensions) could be solved in this way more easily than by an overall optimization technique and certainly with less computer memory. A PDP-8 computer program has been written to solve simple maze problems in this manner and is currently being debugged.

C. Multi-Time Scale Characteristics of Preview Control - Richard A. Miller

Closely related to the previously described problem is the two-time scale preview model shown in Figure 8. The controlled process is of the form $\underline{\dot{x}} = \underline{A}\,\underline{x} + \underline{b}\,u$ with u scalar in the case shown, and the fast-model is a duplicate of the process, but with scaled time constants.

The principle of operation is essentially the same as described above, in that the input is scanned over a future time T allowing an error prediction to be made. If one assumes the fast model gives a good prediction of actual controlled process response, then

$$e(\tau) = y(t_0 + \tau) - z'(\tau) \qquad t_0 < \tau < (T + t_0)$$

is the future error over the interval. For this discussion, consider the importance weighting function as unity for all time.

The hold in the forward loop is the length ζ corresponding to length of time between updates or sampling intervals. The value of the control variable at the sampling instant is then

$$u(t_0 + \zeta) = \int_{t_0}^{t_0 + T} e(\tau)\ \ \omega(\tau)\, d\tau$$

where $\omega(\tau)$ is the preselected weighting function. ω can be made to correspond

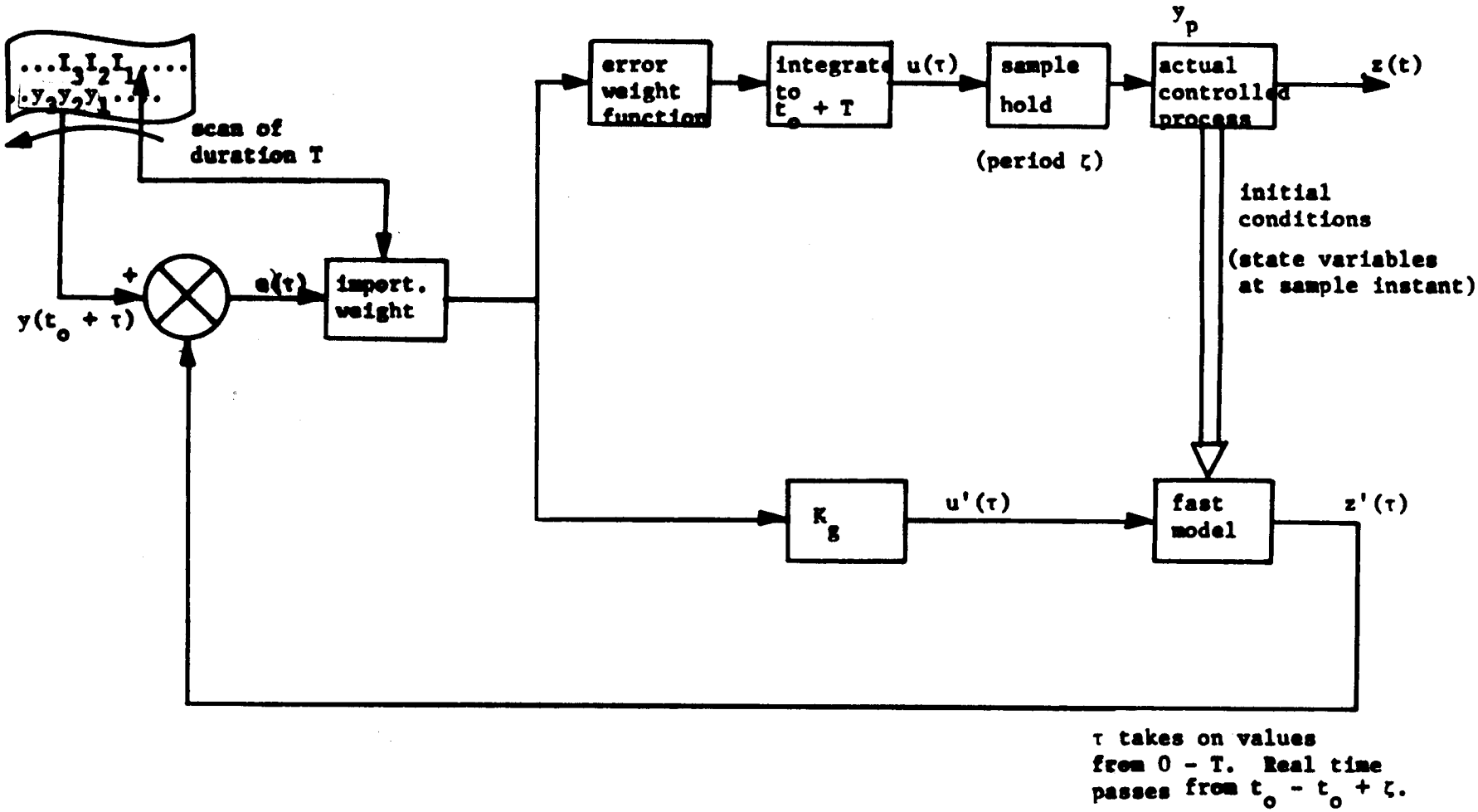


Fig. 8 Two-time scale characterization

to the weighting function in the above theory.

This model is actually a sampled system in that u is discrete. This is the major difference between the two time scale model and the theory above. However, for ζ small the deviation is small, and for corresponding digital simulation there will be negligible difference. If one models the optimal for sampled data u, the resulting control is

$$u(t_0 + \zeta) = \underline{M}^{-1}\, \underline{B}^{-1}\, \underline{\Phi}^T(-\zeta) \int_{t_0+\zeta}^{t_0+T} \underline{\Phi}^T(\sigma, t_0)\, \underline{C}\, \underline{E}\, e(\sigma)d\sigma$$

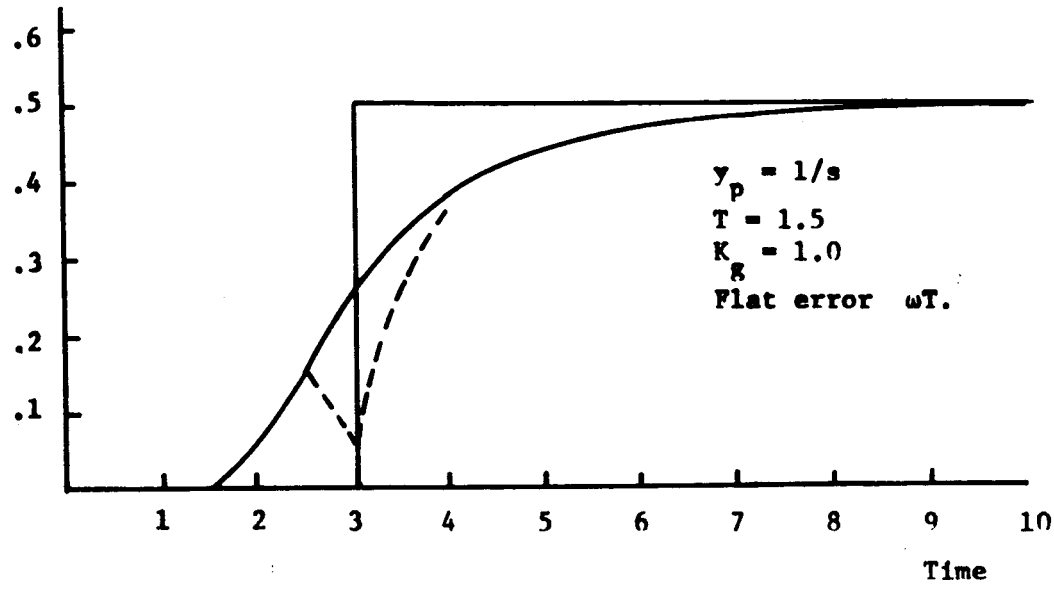
at each sampling interval. The weighting function for the two time scale method can be determined directly from the parameters of the problem,

$$\omega(\sigma) = \underline{M}^{-1}\, \underline{B}^T\, \underline{\Phi}^{\dagger T}(-\zeta)\, \underline{\Phi}^T(\sigma, t_0)\, \underline{C}\, \underline{E}.$$

Experiments have shown that the simple servo model does not predict a future error accurately. A typical run for a plant 1/s is shown in Figure 9. Before the input transient the error nulling servo drives the prediction toward zero, opposite of controlled process response.

In an attempt to improve the quality of error prediction, another model was cascaded creating a three time scale characterization as shown in Figure 10. The method of operation is basically the same, with the second model predicting an error which drives the first model, which in turn drives the process. This is an iterative approach for predicting response. The results for this method are encouraging. As shown below, for 1/s process, the response from two predictors is smooth and follows the general shape of the output. The most encouraging aspect is its consistency. It follows the same pattern, small deviation from actual at start of prediction and diverging at end, as in Figure 11.

The iterative procedure appears to converge rapidly. For example for a 1/s plant the sum integral-squared-error plus integral-squared-control is as follows.



Solid curve - z(t), plant response
Broken curve - $z'(\tau)$ for $t_o = 2.5$.

i.e. Fast model output used to calculate control at $t_o + \zeta$ or t = 2.6. Curve shows poor quality response prediction.

Fig. 9 Typical response for two-time scale technique (one fast model), first order plant.

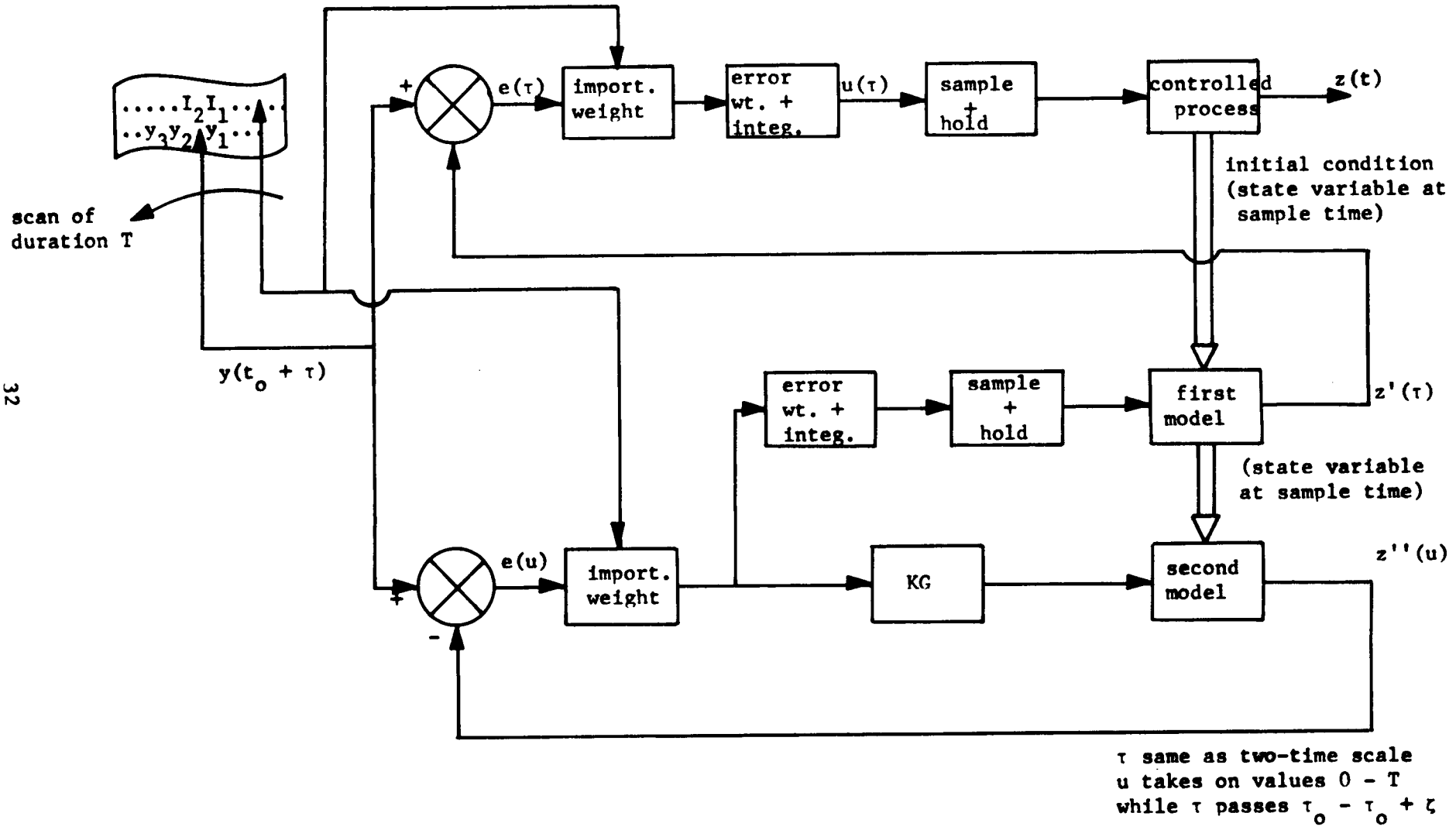
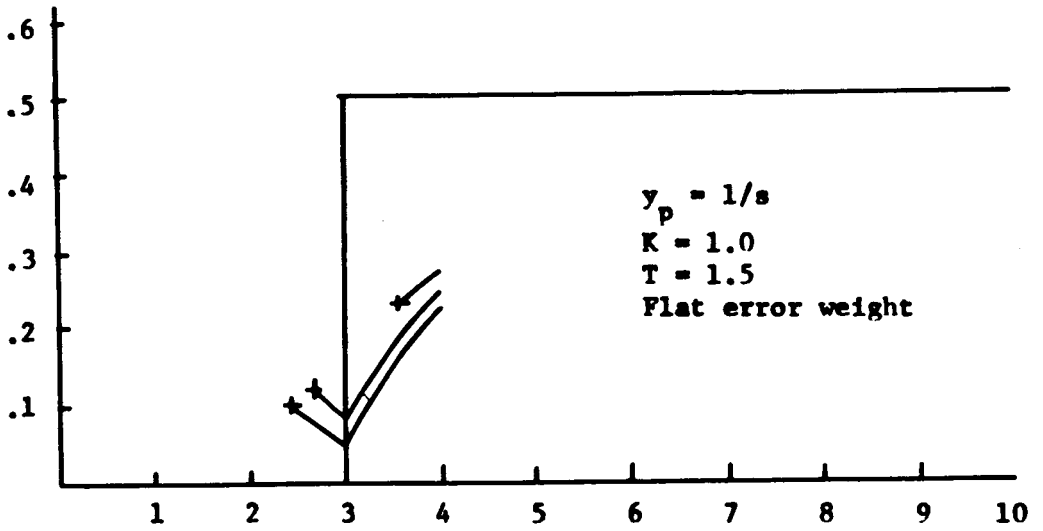


Fig. 10 Three time scale characterization.



second model output

Lower curve - initial time 2.5 used to calculate control for 1st model at time $\tau = 2.6$

Entire sequence generates broken curve below which generates u at t = 2.6 for actual process

Solid curve below is z(t)

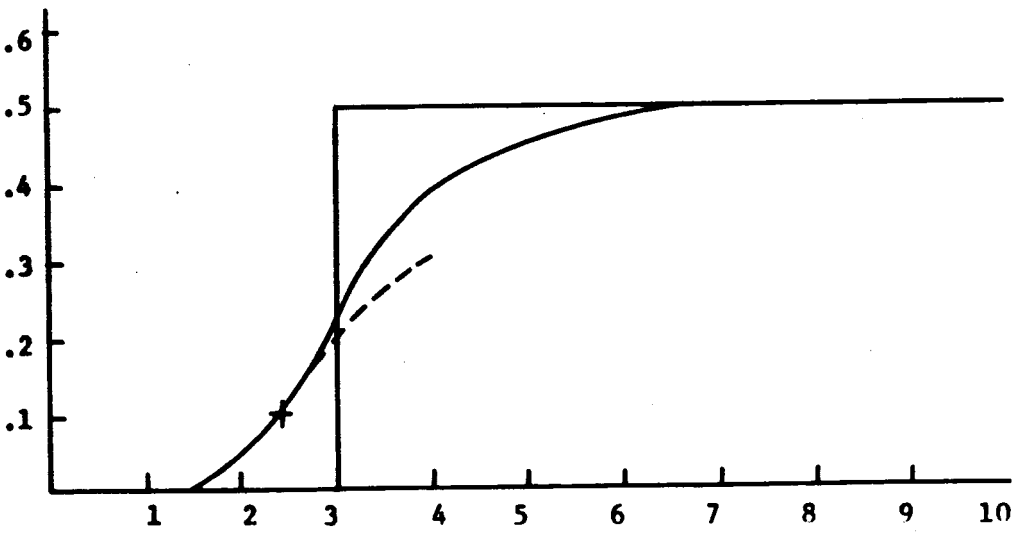


Fig. 11 Response curves for three time scales (two fast models)with first order plant.

No preview	.353950	same preview times and weighting
1 Fast Model	.134020	functions (best results in the 3
2 Models	.132411	cases)

For $1/s(s+1)$ $\int_0^T (e^2 + u^2)d\tau$ (see Fig. 12) is

No preview	.517730	best results for 3 cases
1 Fast Model	.176773	T = 2.5 Flat Er wt.
2 Fast Model	.168847	T = 2.5 op. theory wt. func.

Another iteration is planned to further verify the trend, and to determine the effect on error prediction.

The main disadvantages of this approach is an increase in computation time. If one were merely mechanizing a preview control system, one predictor would be sufficient, but modeling human response requires a reasonable error estimate if valid results are to be obtained.

D. Dynamic Programming and Other Characterizations of Self-Paced Control

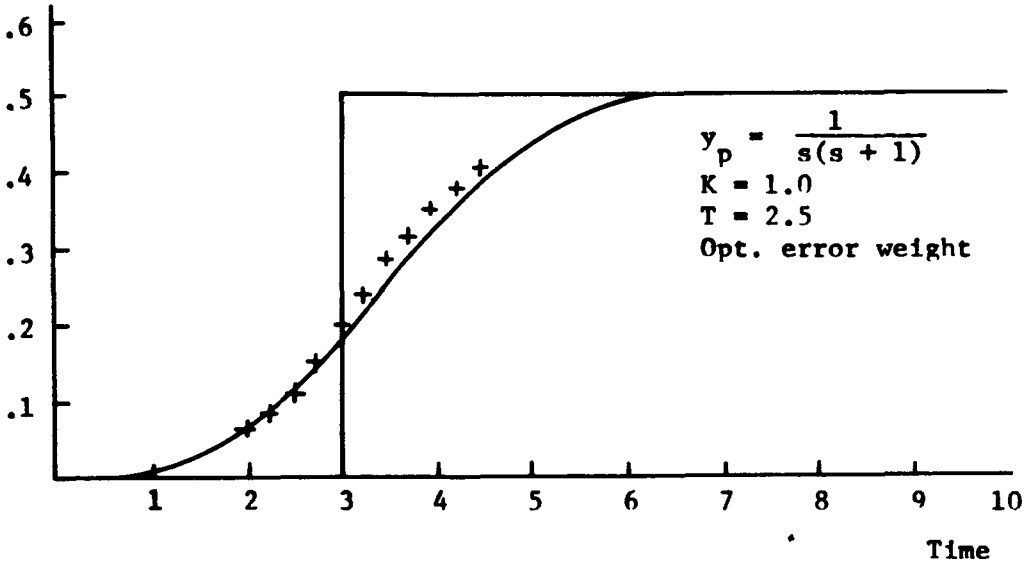
-- Philip A. Hardin

A separate more complete report covering the work described below will be submitted shortly. This report covers two topics, and is divided into two sections. The first is a local variation of Bellman's Dynamic Programming. The second topic is the simulation of a second-order self-paced control system. Self-pacing implies the ability to control velocity in forward as well as lateral direction, where the inputs are given as values of two space coordinates and have no time dependence.

The research for this report was undertaken to provide a background or basis for modeling the human operator as a self-paced system. As such, this report is designed to give investigators some feel for the behavior of self-paced systems as well as algorithms and programs for computing the trajectories of optimal self-paced systems.

Section I

The local variation of the Dynamic Programming algorithm is described.



+ - First model output with t_o = 2.0 - generates u(2.1)

z(t) Actual output

Fig. 12 Response for three time scales (two fast models), second order plant.

Its disadvantages and advantages are discussed in respect to the computing equipment available in 1966. The major advantage is that the grid upon which a continuous system is simulated can be made several orders of magnitude smaller than when using Bellman's Dynamic Programming to simulate the same system. The major disadvantage is that this method is a local method, and is subject to getting trapped in local minima. A procedure for overcoming this difficulty is described.

Section II

A second-order, optimal self-paced control system was simulated using the IBM 7094 computer at the M.I.T. Computation Center. The methods of simulation used were:

a) Bellman's Dynamic Programming

b) a modification of a) that uses the Dynamic Programming algorithm to consider points around a nominal, non-optimal trajectory instead of all points in the state space (a local variation of the regular Dynamic Programming) and

c) a classical gradient analysis.

Methods b) and c) were used to achieve greater resolution than was possible with method a).

The method using Bellman's Dynamic Programming produced satisfactory results, although the grid on which the self-paced system was simulated was very coarse. Method b) increased the grid resolution, and produced better (lower cost) trajectories. Method b) had one weakness in that it could become trapped in local minima. The gradient method of analysis was found to be effective in analyzing only certain types of self-paced systems.